Firewall Configuration for Linux Administrators

Matthew Rossmiller

11/25/03
Firewall Configuration for Linux Administrators

- Review of netfilter/iptables
- Preventing Common Attacks
- Auxiliary Security Programs
- Extended Features
- Management Tools
Review of netfilter/iptables

- kernel structure
- Linux firewall alternatives
- Syntax example
netfilter kernel structure

Kernel Space

Input Hook

Prerouting Hook

Routing Decision

Forward Hook

Forward Traffic

Output Hook

Postrouting Hook

Outbound Traffic

User Space

Network Application

Local traffic

Local traffic

Inbound traffic
# Linux firewall alternatives

<table>
<thead>
<tr>
<th>Firewall</th>
<th>Kernel Version</th>
<th>Packet Filtering</th>
<th>NAT</th>
<th>Connection tracking/Stateful inspection</th>
<th>Application Intelligence</th>
<th>Integrated VPN Capability</th>
<th>Integrated QOS Marking</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipfwadm</td>
<td>1.0</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ipchains</td>
<td>2.2.0</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>netfilter/iptables</td>
<td>2.4.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Partial</td>
<td>X</td>
</tr>
<tr>
<td>Firewall-1</td>
<td>2.4.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
#!/bin/sh
set -e
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin
PATH=$PATH:/usr/sbin:/usr/bin

INET_IF=eth0
LAN_IF=eth1

block_all()
{
    #install kernel modules
    modprobe ip_conntrack
    modprobe ip_conntrack_ftp
    # disable forwarding
    echo 0 > /proc/sys/net/ipv4/ip_forward
    # flush any existing rules
    iptables -F
    iptables --table nat --flush
    iptables -t nat --delete-chain
    # set default policy to be DROP
    iptables -P INPUT DROP
    iptables -P OUTPUT DROP
    iptables -P FORWARD DROP
}

case "$1" in
    start)
        # set firewall to initial (secure) state
        block_all
        # enable loopback interface
        iptables -A INPUT -i lo -p all -j ACCEPT
        iptables -A OUTPUT -o lo -p all -j ACCEPT
        # enable dhcp on the internet interface
        iptables -A INPUT -p UDP -i $INET_IF 
                   --sport 67 --dport 68 -j ACCEPT
        iptables -A OUTPUT -p UDP -o $INET_IF 
                   --dport 67 --sport 68 -j ACCEPT
        # allow forwarding packets for open connections
        iptables -I FORWARD -m state
                   --state INVALID -j DROP
        iptables -I FORWARD -m state
                   --state RELATED,ESTABLISHED -j ACCEPT
        # allow outbound connections from LAN
        iptables -A FORWARD --in-interface $LAN_IF 
                    -j ACCEPT
        # set up NAT to masquerade protected network
        iptables -t nat --append POSTROUTING
                   --out-interface $INET_IF -j MASQUERADE
        # allow web and ftp traffic to the firewall
        iptables -A INPUT -i $INET_IF -p tcp
                   --destination-port ftp -j ACCEPT
        iptables -A INPUT -i $INET_IF -p tcp
                   --destination-port ftp-data -j ACCEPT
        iptables -A INPUT -i $INET_IF -p tcp
                   --destination-port www -j ACCEPT
        # allow all service related outbound traffic
        iptables -A OUTPUT -o $INET_IF -p tcp
                   --source-port ftp -j ACCEPT
        iptables -A OUTPUT -o $INET_IF -p tcp
                   --source-port ftp-data -j ACCEPT
        iptables -A OUTPUT -o $INET_IF -p tcp
                   --source-port www -j ACCEPT
        # allow protected hosts full access to fw
        iptables -A INPUT -i $LAN_IF -j ACCEPT
        iptables -A OUTPUT -o $LAN_IF -j ACCEPT
        #log dropped packets
        iptables -A INPUT -m limit -j LOG
        # enable packets
        iptables -A INPUT -m limit
                   --limit 0/0 -j LOG
        # enable forwarding
        echo 1 > /proc/sys/net/ipv4/ip_forward
        ;;
        stop)
            return firewall to initial (secure) state
            block_all
    ;;
    *) exit
    ;;
    esac
}
Preventing Common Attacks

- Attacks that Firewalls defend against
  - Spoofed Packets
  - Source Routed Traffic
  - DoS attacks *(some protection)*
  - Port Scanning
Spoofed Packets

- Block spoofed packets to prevent SMURF attacks, Redirect attacks and other bad things

- Linux Protection:
  - The kernel can be configured to drop packets whose source address doesn’t route to the inbound interface. (per interface)
    
    ```
    echo 1 > /proc/sys/net/ipv4/conf/${DEVICE}/rp_filter
    echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
    ```

  - Firewall rules can enforce good behavior from internal hosts
    
    ```
    iptables -A FORWARD -i ${INET_DEV} -s ${NETWORK} -j DROP
    ```

  Note: INET_DEV = internet interface
  NETWORK = n.n.n.n/n -- protected network
Source Routed Traffic

- Source routing allows attackers to deliver spoofed packets into a protected network
- Linux Protection:
  - The kernel can be configured to drop source routed packets (per interface)
  - Syntax:
    
    ```bash
    echo 0 > /proc/sys/net/ipv4/conf/${DEVICE}/accept_source_route
    ```
DoS attacks

- Cannot be prevented
- Can prevent DoS effects from reaching into the protected network
- Can block spoofed packets to avoid being the intermediary or originator of a DoS attack
  (See Previous Slide)

- Linux Protection:
  - Firewall can rate-limit inbound traffic
    
    ```
    iptables -A INPUT -p TCP --syn -m limit --limit 5/second -j ACCEPT
    iptables -A INPUT -j DROP
    ```
Port Scanning

- Attackers use port scans to probe the firewall/protected network for vulnerabilities
- Linux protection:
  - PSAD: Port Scan Attack Detector can be configured to sniff firewall logs and report on intrusion fingerprints. Reporting can include automatically blocking attacker addresses, although there are issues with this.
  - In order for psad to work, the firewall should be configured to log all dropped traffic (subject to rate limiting of course)
  - [http://www.cipherdyne.org/psad/manpage.html](http://www.cipherdyne.org/psad/manpage.html)
Auxiliary Security Programs

- VPN
- NAT Masquerade
- Proxy Filtering
- Virus Scanning
Integrated VPN

- VPNs secure communication on a public network
- Linux solution:
  - FreeS/WAN is a GPL IPSec VPN that can be run on the firewall machine to terminate IPSec tunnels into the protected network.
  - IPSec client side tunnel support can be configured for many hosts (even Windoze)
Integrated NAT

- Dynamic Network Address Translation (masquerading) makes it difficult for attackers to access protected hosts
- Linux Solution:
  - `netfilter/iptables` has support for NAT built in
    - `iptables -t nat -A POSTROUTING -s ${NETWORK} -o ${DEVICE} -j MASQUERADE`
Proxy Filtering

- Proxy filtering provides application level security
- Linux Solution:
  - squid web proxy running on the firewall (or protected machine)
  - iptables rules to enforce traffic only flows through the proxy server

  - http://www.squid-cache.org/
Virus Scanning

- A virus scan integrated with sendmail or other MTA protects the network by detecting and removing viruses in individual users’ email traffic

- Linux Solution:
  - Many free and commercial options.
  - Largest is [www.centralcommand.com](http://www.centralcommand.com)
Extended Features

- Load-Balancing
- Modifying TOS
- QOS Queuing
- Policy Routing
Load Balancing

- Using load balancing, we can balance traffic among a number of parallel servers

- Linux Solution:
  - Combined SNAT and DNAT rules to redirect packets to a bank of servers

```bash
iptables -t nat -A POSTROUTING -o ${DEVICE} -j SNAT --to 205.159.243.241
iptables -t nat -A PREROUTING -d ${ADDRESS} -p TCP --dport http -j DNAT --to-dest 10.0.0.2-10.0.0.8
```
Modifying TOS

- Type of Service on packets can be modified to produce improved performance for selected packets (although most routers ignore these bits)

- Linux Solution:
  - Packet mangling with iptables

  **TOS VALUES:**
  Minimize-Delay 16 (0x10)
  Maximize-Throughput 8 (0x08)
  Maximize-Reliability 4 (0x04)
  Minimize-Cost 2 (0x02)
  Normal-Service 0 (0x00)

  **Mangling Example:**
  ```
  iptables -A PREROUTING -t MANGLE -p TCP --sport telnet -j TOS --set-tos 16
  ```
QOS queuing

QOS queuing allows the administrator to prioritize or shape network traffic with more flexibility than the TOS field supports.

Linux support:

- **netfilter/iptables for marking packets**
  ```sh
  iptables -A PREROUTING -i ${DEVICE} -t mangle -p tcp --sport telnet -j MARK --set-mark 1
  ```

- **tc for assigning marked packets various queue**
  ```sh
  tc qdisc add dev ${DEVICE} handle ffff: ingress
tc filter add dev ${DEVICE} parent ffff: protocol ip prio 50 handle 1 fw police rate 1kbit burst 40 mtu : drop flowid :1
  ```

- **[http://lartc.org/](http://lartc.org/)** has a full description of setting up QOS shaping
Policy Routing

Policy Routing assigns alternate routes to packets based on more than destination address lookup

Linux support:

- netfilter/iptables for marking packets
  ```
  iptables -A PREROUTING -i ${DEVICE} -t mangle -p tcp --sport telnet
  -j MARK --set-mark 1
  ```

- iproute2 for enforcing policy routing
  ```
  echo 201 telnet >> /etc/iproute2/rt_tables
  ip rule add fwmark 1 table telnet
  /sbin/ip route add default via ${FAST_NEXTHOP} dev ${FAST_DEVICE} table telnet
  ```

- [http://lartc.org/](http://lartc.org/) describes iproute2 in detail
Management Tools

- fwbuilder
- ipmenu
- IPTables Log Analyzer
fwbuilder

GUI tool for configuring iptables rule sets

http://www-fwbuilder.org/
ipmenu

ASCII (curses) UI tool for configuring iptables rule sets
Also generates policy routing and QOS configuration
http://users.pandora.be/stes/ipmenu.html
IPTables Log Analyzer

- Log file analyzer (to HTML)
- http://www.gege.org/iptables/
Thank You

Questions?